Programming Fundamentals
Lecture #15
Characters and Strings in C Programming

Junaid Hassan

Lecturer CS & IT Department UOS MBDIN

junaidte14@gmail.com

Introduction:

- In this lecture, we'll discuss the C Standard Library functions that facilitate string and
  character processing
- These functions enable programs to process characters, strings, lines of text and blocks of memory

Strings and Characters:

Characters are the fundamental building blocks of source programs. Every program is composed of a sequence of characters that—when grouped together meaningfully—is interpreted by the computer as a series of instructions used to accomplish a task

char ch1 = 'z', ch2 = 'j' etc…

A **string** is a series of characters treated as a single unit. A string may include letters, digits and various special characters such as +, –, *, / and $. **String literals**, or **string constants**, in C are written in double quotation marks

A string in C is an array of characters ending in the null character (‘\0’). A string is accessed via a pointer to the first character in the string. Thus, in C, it is appropriate to say that a string is a pointer—in fact, a pointer to the string’s first character. In this sense, strings are like arrays, because an array is also a pointer to its first element.

String Definition and Initialization:

char color[] = “blue”;
const char *colorPtr = “blue”;

The first definition creates a 5-element array color containing the characters ‘b’, ‘l’, ‘u’, ‘e’ and ‘\0’. The second definition creates pointer variable colorPtr that points to the string “blue” somewhere in memory.

How to get string from user?

```
char  word[20];

scanf(“%19s”, word);
```

Character handling library:

The character-handling library (<ctype.h>) includes several functions that perform useful tests and manipulations of character data.

Prototype	Function description
`int isdigit( int c );`	Returns a true value if c is a digit and 0 (false) otherwise.
`int isalpha( int c );`	Returns a true value if c is a letter and 0 otherwise.
`int isalnum( int c );`	Returns a true value if c is a digit or a letter and 0 otherwise.
`int isxdigit( int c );`	Returns a true value if c is a hexadecimal digit character and 0 otherwise. (See Appendix C, Number Systems, for a detailed explanation of binary numbers, octal numbers, decimal numbers and hexadecimal numbers.)
`int islower( int c );`	Returns a true value if c is a lowercase letter and 0 otherwise.
`int isupper( int c );`	Returns a true value if c is an uppercase letter and 0 otherwise.
`int tolower( int c );`	If c is an uppercase letter, `tolower` returns c as a lowercase letter. Otherwise, `tolower` returns the argument unchanged.
`int toupper( int c );`	If c is a lowercase letter, `toupper` returns c as an uppercase letter. Otherwise, `toupper` returns the argument unchanged.

String Conversion Functions:

This section presents the string-conversion functions from the general utilities library (<stdlib.h>). These functions convert strings of digits to integer and floating-point values. e.g

double d = atof(“99.0”); // converted to double floating value

int i = atoi(“2593”); // converted to int value

long l = atol(“1000000”); // converted to long int value

const char *string = “51.2% are admitted”; /* initialize string */

double d; /* variable to hold converted sequence */
12 char *stringPtr; /* create char pointer */

d = strtod(string, &stringPtr);

Standard Input/Output Library Functions:

This section presents several functions from the standard input/output library (<stdio.h>)
specifically for manipulating character and string data.

```
char sentence[ 80 ]; /* create char array */
printf( “Enter a line of text:\n” );

/* use fgets to read line of text */
fgets( sentence, 80, stdin );
```

Function getchar and puts:

```
1   /* Fig. 8.14: fig08_14.c
2      Using getchar and puts */
3   #include <stdio.h>
4
5   int main( void )
6   {
7      char c; /* variable to hold character input by user */
8      char sentence[ 80 ]; /* create char array */
9      int i = 0; /* initialize counter i */
10
11     /* prompt user to enter line of text */
12     puts( "Enter a line of text:" );
13
14     /* use getchar to read each character */
15     while ( ( c = getchar() ) != '\n') {
16        sentence[ i++ ] = c;
17     } /* end while */
18
19     sentence[ i ] = '\0'; /* terminate string */
20
21     /* use puts to display sentence */
22     puts( "\nThe line entered was:" );
23     puts( sentence );
24     return 0; /* indicates successful termination */
25  } /* end main */
```

String Manipulation Functions:

The string-handling library (<string.h>) provides many useful functions for manipulating string data (copying strings and concatenating strings), comparing strings, searching strings for characters and other strings, tokenizing strings (separating strings into logical pieces) and determining the length of strings. This section presents the string-manipulation functions of the string-handling library

Function prototype	Function description
`char *strcpy( char *s1, const char *s2 )`	Copies string `s2` into array `s1`. The value of `s1` is returned.
`char *strncpy( char *s1, const char *s2, size_t n )`	Copies at most `n` characters of string `s2` into array `s1`. The value of `s1` is returned.
`char *strcat( char *s1, const char *s2 )`	Appends string `s2` to array `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned.
`char *strncat( char *s1, const char *s2, size_t n )`	Appends at most `n` characters of string `s2` to array `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned.

String Concatenation Functions:

```
1  /* Fig. 8.19: fig08_19.c
2     Using strcat and strncat */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main( void )
7  {
8     char s1[ 20 ] = "Happy "; /* initialize char array s1 */
9     char s2[] = "New Year "; /* initialize char array s2 */
10    char s3[ 40 ] = ""; /* initialize char array s3 to empty */
11
12    printf( "s1 = %s\ns2 = %s\n", s1, s2 );
13
14    /* concatenate s2 to s1 */
15    printf( "strcat( s1, s2 ) = %s\n", strcat( s1, s2 ) );
16
17    /* concatenate first 6 characters of s1 to s3. Place '\0'
18       after last character */
19    printf( "strncat( s3, s1, 6 ) = %s\n", strncat( s3, s1, 6 ) );
20
21    /* concatenate s1 to s3 */
22    printf( "strcat( s3, s1 ) = %s\n", strcat( s3, s1 ) );
23    return 0; /* indicates successful termination */
24 } /* end main */
```

String Comparison Functions:

Function prototype	Function description
`int strcmp( const char *s1, const char *s2 );`	Compares the string `s1` with the string `s2`. The function returns `0`, less than `0` or greater than `0` if `s1` is equal to, less than or greater than `s2`, respectively.
`int strncmp( const char *s1, const char *s2, size_t n );`	Compares up to `n` characters of the string `s1` with the string `s2`. The function returns `0`, less than `0` or greater than `0` if `s1` is equal to, less than or greater than `s2`, respectively.

String Search Functions:

```
   Using strchr */
#include <stdio.h>
#include <string.h>

int main( void )
{
   const char *string = "This is a test"; /* initialize char pointer */
   char character1 = 'a'; /* initialize character1 */
   char character2 = 'z'; /* initialize character2 */

   /* if character1 was found in string */
   if ( strchr( string, character1 ) != NULL ) {
      printf( "\'%c\' was found in \"%s\".\n",
         character1, string );
   } /* end if */
   else { /* if character1 was not found */
      printf( "\'%c\' was not found in \"%s\".\n",
         character1, string );
   } /* end else */

   /* if character2 was found in string */
   if ( strchr( string, character2 ) != NULL ) {
      printf( "\'%c\' was found in \"%s\".\n",
         character2, string );
   } /* end if */
   else { /* if character2 was not found */
      printf( "\'%c\' was not found in \"%s\".\n",
         character2, string );
   } /* end else */

   return 0; /* indicates successful termination */
} /* end main */
```